IN THE CLAIMS:

Please cancel claims 4, 7, 10, 14-20, 24, and 32-42 and amend the claims as follows:

1. (Currently Amended) A computer-implemented method for parallel processing ~~of~~ a sequence of requests received by an application server configured to execute a plurality of threads, comprising:

initiating, by the application server, a first thread ~~primary executing entity~~ configured to (i) perform the sequence of requests received from a user interacting with a web-based application and (ii) to maintain state information specific to the first thread ~~primary executing entity~~, wherein each request, of the sequence, is composed as a uniform resource locater submitted to the web-based application;

initiating, by the application server, a second thread ~~secondary executing entity~~ configured to perform the sequence of requests and to maintain state information specific to the second thread ~~secondary executing entity~~;

performing, sequentially, by the ~~primary executing entity~~ first thread, the sequence of requests; and

performing, sequentially, by the second thread ~~secondary executing entity~~, the same sequence of requests, wherein the second thread performs requests, of the sequence, previously performed by the ~~primary executing entity~~ first thread ~~in~~ a specified number of steps behind first thread ~~time-delayed and step-wise fashion~~ and while the ~~primary executing entity~~ first thread continues to execute requests, from the sequence of requests, whereby ~~each executing entity~~ the first thread and second thread each independently maintain~~s its own respective~~ state information ~~independent of, and temporally displaced from, the other executing entity~~ regarding results of each request, of the sequence, performed by the first thread and the second thread, respectively.

2. (Currently Amended) The method of claim 1, ~~further comprising:~~

~~making the~~ wherein the results of performing the sequence of requests by the first thread is ~~performance of the primary executing entity~~ visible to a user; and

~~making the~~ wherein the results of performing the sequence of requests by the second thread is ~~performance of the secondary executing entity~~ transparent to the user.

3. (Currently Amended) The method of claim 1, further comprising:

upon detecting an error in the results from performing, by the first thread, one of the sequence of requests, terminating the first thread ~~primary executing entity~~; and ~~then~~

performing, by the second thread, ~~secondary executing entity~~, at least a portion of the sequence of requests performed by the terminated first thread ~~primary executing entity~~ and not ~~previously~~ yet performed by the second thread ~~secondary executing entity~~.

4. (Cancelled)

5. (Currently Amended) The method of claim 1, further comprising:

upon completion of the sequence of requests by the first thread, producing ~~output~~ a primary result for the sequence of requests ~~by the primary executing entity~~;

subsequently, upon completion of the sequence of requests by the second thread, producing a secondary result for the sequence of requests ~~output by the secondary executing entity~~;

displaying the ~~output~~ the primary result produced by the first thread ~~primary executing entity~~; and

discarding, without displaying, the secondary result ~~output~~ produced by the second thread ~~secondary executing entity~~.

6. (Currently Amended) The method of claim 1, further comprising,

upon detecting ~~encountering~~ an error in the results from performing, by the first thread, one of the sequence of requests ~~primary executing entity~~;

terminating execution of the first thread ~~primary executing entity~~; and

~~recovering from the error by~~ returning ~~a~~ to the user ~~to~~ an indication of the request being ~~handled~~ performed by the first thread resulting in the detected error ~~primary executing entity when being terminated~~

7. (Cancelled)

8. (Currently Amended) The method of claim 1, further comprising, ~~upon encountering an error by the primary executing entity:~~

upon detecting an error in the results from performing, by the first thread, one of the sequence of requests:

terminating the first thread ~~primary executing entity~~; and

recovering from the error by executing remaining requests of the sequence by the second thread up to, but not including, the request resulting in the detected error ~~returning a user to a request within a range of requests between a request being handled by the secondary executing entity and a request being handled by the primary executing entity at the time of encountering the error;~~

prompting the user to modify the request which resulted in the detected error;

receiving a modification to the request;

executing the modified request;

executing any remaining requests, of the sequence of requests; and

displaying a final result produced by the second thread from executing (i) the sequence of requests up to, but not including, the request resulting in the detected error, (ii) the modified request and (iii) the remaining requests of the sequence, if any.

9. (Currently Amended) The method of claim 1, wherein the requests of the sequence are time ordered and processed by each of the first thread and the second thread ~~performing executing entity~~ according to the time order.

10. (Cancelled)

11. (Currently Amended) A computer-implemented method for parallel processing of requests received by an application server configured to execute a plurality of threads, comprising:

receiving a sequence of ~~user~~ requests from a user, wherein each request, of the sequence, is composed as a uniform resource locater submitted to the web-based application;

placing the ~~user~~ sequence of requests on a queue managed by the application server in a time-ordered manner;

performing, by a first thread managed by the application server ~~primary executing entity~~, each ~~current user~~ request, of the sequence, upon being placed on the queue; and

performing, by a second thread managed by the application server ~~secondary executing entity~~, at least a portion of the user requests on the queue step-wise with the first thread ~~primary executing entity~~ and N-requests behind the first thread ~~primary executing entity~~; wherein ~~each of the executing entities~~ maintain their own respective state information the first thread and second thread each independently maintain~~s~~ ~~their own respective~~ state information regarding results of each request, of the sequence, performed by the first thread and the second thread, respectively.

12. (Currently Amended) The method of claim 11, further comprising:

upon detecting ~~encountering~~ an error in the results from performing, by the first thread, one of the sequence of requests ~~primary executing entity~~;

terminating execution of the first thread ~~primary executing entity~~; and

~~recovering from the error by~~ returning ~~a~~ to the user ~~to~~ an indication of the request ~~being handled~~ performed by the first thread which resulted in the detected error ~~primary executing entity when being terminated~~.

13 - 20. (Cancelled)

21. (Currently Amended) A computer readable storage medium containing a program which, when executed, implements an operation for parallel processing ~~of~~ a sequence of requests received by an application server configured to execute a plurality of threads, the operation comprising:

initiating, by the application server, a first thread ~~primary executing entity~~ configured to (i) perform the sequence of requests received from a user interacting with a web-based application and (ii) to maintain state information specific to the first thread ~~primary executing entity~~, wherein each request, of the sequence, is composed as a uniform resource locater submitted to the web-based application;

initiating, by the application server, a second thread ~~secondary executing entity~~ configured to:

perform requests, from the sequence of requests, previously performed by the first thread ~~primary executing entity in~~ a specified number of steps behind first thread ~~time-delayed and step-wise fashion~~ and while the first thread ~~primary executing entity~~ continues to execute requests, of the sequence; and

maintain state information specific to the second thread ~~secondary executing entity~~, whereby the first thread and the second thread each ~~executing entity~~ maintain~~s its own respective~~ state information ~~independent of, and temporally displaced from, the other executing entity~~ regarding results of each request, of the sequence, performed by the first thread and the second thread, respectively.

22. (Currently Amended) The computer readable storage medium of claim 21, ~~the operation further comprising~~:

~~making the~~ wherein the results of performing the sequence of requests by the first thread is ~~performance of the primary executing entity~~ visible to a user; and

~~making the~~ wherein the results of performing the sequence of requests by the second thread is ~~performance of the secondary executing entity~~ transparent to the user.

23. (Currently Amended) The computer readable storage medium of claim 21, the operation further comprising:

upon detecting an error in the results from performing, by the first thread, one of the sequence of requests, terminating the first thread ~~primary executing entity~~; and ~~then~~

performing, by the second thread, ~~secondary executing entity~~, one or more requests of the sequence previously performed by the terminated first thread ~~primary executing entity~~ and not ~~previously~~ yet performed by the secondary executing entity.

24. (Cancelled)

25. (Currently Amended) The computer readable storage medium of claim 21, wherein the operation further comprises ~~further comprising~~:

upon completion of the sequence of requests by the first thread, producing ~~output~~ a primary result for the sequence of requests ~~by~~ ~~the~~ ~~primary executing entity~~;

subsequently, upon completion of the sequence of requests by the second thread, producing a secondary result for the sequence of requests ~~output by the secondary executing entity~~;

displaying the ~~output~~ the primary result produced by the first thread ~~primary executing entity~~; and

discarding, without displaying, the secondary result ~~output~~ produced by the second thread ~~secondary executing entity~~.

26. (Currently Amended) The computer readable storage medium of claim 21, ~~further comprising,~~ wherein the operation further comprises:

upon detecting ~~encountering~~ an error in the results from performing, by the first thread, one of the sequence of requests ~~primary executing entity~~;

terminating execution of the first thread ~~primary executing entity~~; and

~~recovering from the error by~~ returning ~~a~~ to the user ~~to~~ an indication of the request ~~currently being handled~~ performed by the first thread which resulted in the detected error ~~primary executing entity immediately prior to being terminated~~.

27. (Currently Amended) The computer readable storage medium of claim 21, wherein the requests of the sequence are time ordered and processed by each of the first thread and the second thread ~~performing executing entity~~ according to the time order.

28. (Currently Amended) The computer readable storage medium of claim ~~27~~ 21, ~~further comprising~~ wherein the operation further comprise:

upon detecting an error in the results from performing, by the first thread, one of the sequence of requests

terminating the first thread ~~primary executing entity~~; and ~~then~~

performing, by the second thread ~~secondary executing entity~~, remaining requests of the sequence up to, but not including, the request resulting in the detected error ~~the requests performed by the terminated primary executing entity between a last request handled by the terminated primary executing entity and a last request handled by the secondary executing entity~~;

prompting the user to modify the request which resulted in the detected error;

receiving a modification to the request;

executing the modified request;

executing any remaining requests, of the sequence of requests; and

displaying a final result produced by the second thread from executing (i) the sequence of requests up to, but not including, the request resulting in the detected error, (ii) the modified request and (iii) the remaining requests of the sequence, if any.

29. (Currently Amended) A computer readable storage medium containing a program which, when executed, implements an operation for parallel processing ~~of~~ requests received by an application server configured to execute a plurality of threads, the operation comprising:

receiving a sequence of ~~user~~ requests from a user, wherein each request, of the sequence, is composed as a uniform resource locater submitted to the web-based application;

placing the ~~user~~ sequence of requests on a queue managed by the application server in a time-ordered manner;

performing, by a first thread managed by the application server ~~primary executing entity~~, each ~~current user~~ request, of the sequence, upon being placed on the queue; and

performing, by a ~~secondary executing entity~~, at least a portion of the user requests on the queue step-wise with the ~~primary executing entity~~ and N-requests behind the ~~primary executing entity~~; wherein ~~each of the executing entities~~ the first thread and second thread each independently maintain~~s~~ ~~their own respective~~ state information regarding results of each request, of the sequence, performed by the first thread and the second thread, respectively.

30. (Currently Amended) The computer readable storage medium of claim 29, ~~further comprising,~~ wherein the operation further comprises:

upon detecting ~~encountering~~ an error in the results from performing, by the first thread, one of the sequence of requests ~~primary executing entity~~;

terminating execution of the first thread ~~primary executing entity~~; and

~~recovering from the error by~~ returning ~~a~~ to the user ~~to~~ an indication of the request ~~currently being handled~~ performed by the first thread which resulted in the detected error ~~primary executing entity immediately prior to being terminated~~

31 – 42. (Cancelled)